

BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	

```
BBBBBBBBB  TTTTTTTTTT MM      MM EEEEEEEEEEE MM      MM 8888888 SSSSSSSSS SSSSSSSSS
BBBBBBBBB  TTTTTTTTTT MM      MM EEEEEEEEEEE MM      MM 8888888 SSSSSSSSS SSSSSSSSS
BB      BB  TT      MM      MM M      M      EE      EE MM      MM 88      88 SS      SS
BB      BB  TT      MM      MM M      M      EE      EE MM      MM 88      88 SS      SS
BB      BB  TT      MM      MM M      M      EE      EE MM      MM 88      88 SS      SS
BB      BB  TT      MM      MM M      M      EE      EE MM      MM 88      88 SS      SS
BBBBBBBBB  TT      MM      MM EEEEEEEEEEE MM      MM 8888888 SSSSSSS SSSSSSS
BBBBBBBBB  TT      MM      MM EEEEEEEEEEE MM      MM 8888888 SSSSSSS SSSSSSS
BB      BB  TT      MM      MM EE      EE MM      MM 88      88 SS      SS
BB      BB  TT      MM      MM EE      EE MM      MM 88      88 SS      SS
BB      BB  TT      MM      MM EE      EE MM      MM 88      88 SS      SS
BB      BB  TT      MM      MM EE      EE MM      MM 88      88 SS      SS
BBBBBBBBB  TT      MM      MM EEEEEEEEEEE MM      MM 8888888 SSSSSSSSS SSSSSSSSS
BBBBBBBBB  TT      MM      MM EEEEEEEEEEE MM      MM 8888888 SSSSSSSSS SSSSSSSSS
```

```
LL      SSSSSSSS
LL      SSSSSSSS
LL      SS
LL      SS
LL      SS
LL      SS
LL      SSSSSS
LL      SSSSSS
LL      SS
LL      SS
LL      SS
LL      SS
LLLLLLLLL SSSSSSSS
LLLLLLLLL SSSSSSSS
```


(2) 119
(3) 148

Declarations
CHECKMEM_8SS, Identify 11/8SS memory

```
0000 1 .TITLE BTMEM8SS - Configure and Test 11/8SS Memory
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *****
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27 *****
0000 28 ++
0000 29
0000 30 FACILITY:
0000 31
0000 32 Linked with VMB.EXE - part of the
0000 33 bootstrap module for VAX 11/8SS hardware.
0000 34
0000 35 ENVIRONMENT:
0000 36
0000 37 Runs at IPL 31, kernel mode, memory management is OFF, IS=1
0000 38 (running on interrupt stack), and code must be PIC.
0000 39
0000 40
0000 41 FUNCTIONAL DESCRIPTION:
0000 42
0000 43 This routine is 11/8SS specific and
0000 44 determines how many memory controllers are on the system,
0000 45 where they are, how much memory they control, which pages
0000 46 of that memory are present (and good). Then the routines
0000 47 set bits in the PFN bitmap to represent each present (and
0000 48 good) page of memory.
0000 49
0000 50 As a side effect, the routines store the type of adapter located
0000 51 at each bus slot in the RPB.
0000 52
0000 53 INPUTS:
0000 54
0000 55 R7 - address of the SCB
0000 56 R11 - address of the RPB
0000 57
```


0000 58 : IMPLICIT INPUTS:
0000 59 :
0000 60 : The positions on the 11/8SS system bus, the BI, are called nodes, and
0000 61 : are identified by node numbers 0-15.
0000 62 :
0000 63 : BI I/O space begins at ^x20000000 (512MB), and each node occupies
0000 64 : an 8KB node space of registers.
0000 65 :
0000 66 : BI nodes are self identifying in that the first register (relative 0)
0000 67 : in the node space is constrained to be the DEVTYPE register. The
0000 68 : format of this register is as follows:
0000 69 :
0000 70 :

31	16	15	14	8	7	0
Revision Code			N	MBZ for		
			D	memory		
			Device Type			

0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 : This register is to be interpreted as follows:
0000 80 :
0000 81 : The Device Type field is broken up into three sub-fields.
0000 82 : The main sub-field of interest here is the one defined
0000 83 : by bits <14:08>. If this sub-field is zero then this
0000 84 : node is a BI memory node. All other types of node,
0000 85 : i.e. processors, adapters, etc., are constrained to
0000 86 : have non-zero values here.
0000 87 :
0000 88 : The ND field, bit <15>, if non-zero, indicates that the
0000 89 : node is a non-DEC device.
0000 90 :
0000 91 :
0000 92 : OUTPUTS:
0000 93 :
0000 94 : R7, R8, R11, and SP are preserved
0000 95 : All others (including AP and FP) are altered
0000 96 :
0000 97 : IMPLICIT OUTPUTS:
0000 98 :
0000 99 : The PFN bitmap is modified to describe all of physical memory.
0000 100 :
0000 101 : RPB\$\$_PFNCNT stores the number of pages of physical memory.
0000 102 :
0000 103 :
0000 104 : AUTHOR:
0000 105 :
0000 106 : Robert L. Rappaport, creation date 10-Oct-1983
0000 107 :
0000 108 : REVISION HISTORY:
0000 109 :
0000 110 : V03-002 RLR0001 Robert L. Rappaport 6-Apr-1984
0000 111 : Several modifications. 1) Test for BROKE bit, and do NOT
0000 112 : test memory if set; 2) Clear error bits in CSR1 and CSR2;
0000 113 : 3) move replacement of Machine Check handler from loop;
0000 114 : 4) PUSHHL original Machine Check handler addr and POPL it

BTMEM8SS
V04-000

- Configure and Test 11/8SS Memory G 11

15-SEP-1984 23:44:25 VAX/VMS Macro V04-00
4-SEP-1984 23:03:23 [BOOTS.SRC]BTMEM8SS.MAR;1

Page 3
(1)

0000 115 :
0000 116 :
0000 117 :--

after loop.


```
0000 119      .SBTTL  Declarations
0000 120
0000 121      .DEFAULT DISPLACEMENT, WORD
0000 122
0000 123      :
0000 124      : Macros to describe VMS data structures
0000 125      :
0000 126
0000 127      $BIICDEF      ; BIIC registers
0000 128      $BIMEMDEF    ; BI Memory registers
0000 129      $DMPDEF      ; System dump file header definitions
0000 130      $IO8SSDEF    ; 11/8SS definitions
0000 131      $NDTDEF      ; Nexus device types
0000 132      $PRDEF       ; Processor registers
0000 133      $PR8SSDEF    ; CPU Specific Processor registers
0000 134      $RPBDEF      ; Restart parameter block
0000 135      $BUADEF      ; BI-UNIBUS adapter
0000 136
0000 137      :
0000 138      : Macros
0000 139      :
0000 140
0000 141      .MACRO  ERROR,STR      ; Outputs an error string to the
0000 142      BSBW      ERROUT      ; console terminal.
0000 143      .ASCIZ  STR
0000 144      .ENDM   ERROR
0000 145
00000000 146      .PSECT  YBTMEM, LONG
```

```
0000 148      .SBTTL CHECKMEM_8SS, Identify 11/8SS memory
0000 149
0000 150      :++
0000 151      :
0000 152      : CHECKMEM_8SS, Locate and test memory for 11/8SS
0000 153      :
0000 154      :--
0000 155
0000 156 CHECKMEM_8SS::                                ; Entry for 11/8SS.
0000 157
0000 158      :
0000 159      : Start testing slot positions to find adapters. First save the stack
0000 160      : position so it can be restored after a machine check.
0000 161      :
0000 162
0000 163 INIT_SEARCH:                                    ; Start searching for adapters
0000 164
0000 165      : Save original machine check handler address on stack.
0000 166
0000 167      PUSHL 4(R7)                                ; Save original handler address.
0000 168      MOVL SP,FP                                  ; Save current top of stack.
0000 169
0000 170      :
0000 171      : Set up the physical address of the 1st slot on the system bus and
0000 172      : the address of the adapter type table.
0000 173      :
0000 174
0000 175      MOVAB @#IO8SS$AL_IOBASE,R4                ; Get address of 1st slot.
0000 176      MOVZBL #IO8SS$AL_NNEX,AP                  ; Set up NEXUS loop counter.
0000 177
0000 178      :
0000 179      : During this memory locate and test loop, the following registers are
0000 180      : used:
0000 181      :
0000 182      : R0      - the contents of the slot's configuration register
0000 183      : R4      - address of the configuration register at the current
0000 184      :           slot position
0000 185      : R7      - address of the SCB
0000 186      : R9      - bit setting in memory present map;
0000 187      :           starting page number in this controller
0000 188      : R10     - address of the memory description list in RPB (pagcnt & pfn)
0000 189      : R11     - address of the RPB
0000 190      :
0000 191      : Initialize R10 (RPB memory descriptor list pointer) for search loop
0000 192      :
0000 193      MOVAB RPB$MEMDSC(R11),R10                    ; Set pointer to memory description list
0000 194      BICL  #<RPB$M_MPM ! RPB$M_USEMPM ! RPB$M_FINDMEM>, -
0000 195      RPB$B_OOTR5(R11)                            ; Clear all MA780-specific boot flags
0000 196
0000 197
0000 198 TRY_NEXUS_8SS:                                    ; Memory locate and test loop.
0000 199
0000 200      : Set up a machine check fault handler to gain control if the main loop
0000 201      : here addresses a non-existent configuration register (an empty slot).
0000 202      : Note we do this within the loop since we re-set to another machine
0000 203      : check handler later on in the loop.
0000 204
```

04 A7 DD 0000 167
SD 5E D0 0003 168
0006 169
0006 170
0006 171
0006 172
0006 173
54 20000000 9F 9E 0006 175
5C 10 9A 000D 176
0010 177
0010 178
0010 179
0010 180
0010 181
0010 182
0010 183
0010 184
0010 185
0010 186
0010 187
0010 188
0010 189
0010 190
0010 191
0010 192
30 AB 5A 00BC CB 9E 0010 193
00005800 8F CA 0015 194
001D 195
001D 196
001D 197
001D 198
001D 199
001D 200
001D 201
001D 202
001D 203
001D 204


```
04 A7 6D'AF 9E 001D 205      MOVAB  B*DO_NEXT_8SS+1,4(R7)  ; Set up fault handler (+1 for
0022 206                      ; handler execution on the
0022 207                      ; interrupt stack).
0022 208
0022 209      ; Then read the slot's configuration register.
0022 210
50 64 D0 0022 211      MOVL   (R4),R0                      ; Read CR at current slot.
0025 212
0025 213      ;
0025 214      ; Execution continues here if the configuration register is present.
0025 215      ; If the adapter type is a memory controller, proceed to test memory.
0025 216      ; Otherwise, move to the next BI slot.
0025 217      ;
0025 218
0025 219
50 50 08 EF 0025 220      EXT2V  #BIICSV_MEMNODE,-          ; Extract sub-field of BI devtype
07 07 07 12 0027 221      #BIICSS_MEMNODE,R0,R0        ; field that tells if we have memory.
40 40 40 12 002A 222      BNEQ   DO_NEXT_8SS              ; No, advance to next slot.
002C 223
002C 224      ; Here check BROKE bit to see if memory passed self test.
002C 225
50 0100 C4 D0 002C 226      MOVL   BIMEM$L_CSR1(R4),R0        ; Read CSR containing BROKE bit.
0C 0C E0 0031 227      BBS     #BIMEM$V_BROKE,-          ; If BROKE, ignore this node.
37 50 37 50 0033 228      R0,DO_NEXT_8SS
0035 229
0035 230      ; Memory controller found:
0035 231
50 20 A4 D0 0035 232      MOVL   BIIC$L_SAR(R4),R0          ; R0 = Starting Address for this memory.
51 24 A4 D0 0039 233      MOVL   BIIC$L_EAR(R4),R1          ; R1 = Ending Address for this memory.
59 50 17 9C 003D 234      ROTL   #<32-9>,R0,R9              ; Equivalent of ASHL #-9,R0,R9. Result
0041 235                      ; is starting PFN to R9.
53 51 50 C3 0041 236      SUBL3  R0,R1,R3                  ; R3 = length of memory segment in bytes.
53 53 17 9C 0045 237      ROTL   #<32-9>,R3,R3              ; Equivalent of ASHL #-9,R3,R3. Result
0049 238                      ; is # of pages to R3.
0049 239
8A 53 D0 0049 240      MOVL   R3,(R10)+                    ; Save # of pages in this memory
004C 241      ASSUME  DMP$V_TR EQ 24
004C 242      ASSUME  DMP$S_TR EQ 8
FF AA 10 90 004C 243      MOVB   #108SS$AL_NNEX,-1(R10)    ; Compute the TR number for this
FF AA 5C 82 0050 244      SUBB   AP,-1(R10)                ; memory and store in descriptor
8A 59 D0 0054 245      MOVL   R9,(R10)+                    ; Save starting PFN for this memory
0057 246
0057 247      ;
0057 248      ; Before starting memory test, establish a page skipping handler for
0057 249      ; machine checks, and turn off the cache so that writes followed by
0057 250      ; reads to memory don't write to memory and then read from the cache.
0057 251      ; Also, enable CRD error reporting if requested by the RPB BOOTR5 flag.
0057 252      ;
0057 253
04 A7 00B9'CF 9E 0057 254      MOVAB  PAGE_MCHECK_8SS+1,4(R7) ; Set page skipping handler (+1
005D 255                      ; for interrupt stack).
01 01 DA 005D 256      MTPR    #PR8SS$M_CADR_D,-          ; Turn off memory cache.
25 25 25 005F 257      #PR8SS$ _CADR
52 0088'CF 9E 0060 258      MOVAB  TEST_QUAD_8SS,R2          ; Page test routine address
FF98' 30 0065 259      BSBW    BOOT$TEST_MEM          ; Test the specified range of PFN's
02 02 11 0068 260      BRB     DO_NEXT_8SS              ; Do the next controller if any
006A 261
```

```
006A 262 .ALIGN LONG ; Longword-aligned handler.
006C 263
006C 264
006C 265 : Fault handler for non-existent configuration register, or unreadable
006C 266 : registers, or a non-memory controller slot device. Restore stack
006C 267 : pointer, clear all errors, and try for another slot if any remain.
006C 268
006C 269
006C 270 DO_NEXT_8SS: ; Skip to next slot.
006C 271 MOVL FP,SP ; Restore stack pointer.
26 FFFF FFFF 8F DA 006F 272 MTPR #-1, #PRS_MCESR ; Clear any faults.
54 2000 C4 9E 0076 273 MOVAB IO8SS$AL_PERNEX(R4),R4 ; Move to next slot.
9F 5C F5 007B 274 SOBGTR AP,TRY_NEXUS_8SS ; If still a slot, loop.
007E 275
007E 276
007E 277 : Reestablish the normal machine check fault handler.
007E 278
007E 279
007E 280 ASSUME PR8SS$V CADR D EQ 0
25 00 DA 007E 281 MTPR #0, #PR8SS$_CAADR ; Re-enable cache.
04 A7 8E D0 0081 282 POPL 4(R7) ; Restore original handler address.
8A D4 0085 283 CLRL (R10)+ ; Indicate end of RPB memory descr list
05 0087 284 RSB ; Return to main routine.
0088 285
0088 286 :++
0088 287
0088 288 : Functional Description:
0088 289
0088 290 : Test a page of 8SS memory.
0088 291
0088 292 : Calling Sequence:
0088 293
0088 294 : JSB TEST_QUAD_8SS
0088 295
0088 296 : Inputs:
0088 297
0088 298 : R0 = starting address to test
0088 299 : R1 = Quad word iteration count (64)
0088 300 : R4 => Node Registers
0088 301 : R11= Address of RPB
0088 302
0088 303 : Outputs:
0088 304
0088 305 : Returns via RSB if the entire page is OK
0088 306 : Error exit via Machine Check code to BOO$PAGE_MCHECK
0088 307
0088 308 :--
0088 309
0088 310 TEST_QUAD_8SS: ; Test 1 quadword at a time.
80 60 7C 0088 311 CLRQ (R0) ; Clear a quadword.
80 80 D1 008A 312 CMPL (R0)+,(R0)+ ; Read both longwords, and
008D 313 ; advance to next quadword.
008D 314
008D 315
008D 316 : If no gross errors occur in the clear to the quadword or in the
008D 317 : subsequent read instruction, then execution continues below. Otherwise
008D 318 : execution goes to the fault handler.
```



```
008D 319 ;
008D 320 ;
51    F8 51    F5 008D 321 SOBGTR R1,TEST_QUAD_8SS ; Continue clearing unless done.
0100 C4 51    D0 0090 322 MOVL   BIMEMSL_CSR1(R4),R1 ; Read memory CSR1.
51    0104 C4  D0 0095 323 MOVL   R1,BIMEMSL_CSR1(R4) ; Re-write (write 1 to clear) CSR1.
0104 C4 51    D0 009A 324 MOVL   BIMEMSL_CSR2(R4),R1 ; Read memory CSR2 and retain value
0104 C4 51    D0 009F 325 ; in R1 for later testing CRD bit.
0104 C4 51    D0 009F 326 MOVL   R1,BIMEMSL_CSR2(R4) ; Re-write (write 1 to clear) CSR2.
01 30 AB 10    E1 00A4 327 BBC     #RPBSV CRDTEST, - ; Branch if CRD test is requested.
01 30 AB 05    00A6 328 RSB     RPBSL_BOOTR5(R11),10$
00AA 329 5$:
00AA 330 ;
00AA 331 ; Check if a CRD error occurred on this page.
00AA 332 ;
00AA 333 10$:
51    20000000 8F D3 00AA 334 BITL   #BIMEMSL_CRDLOGR,R1 ; Test if CRD errors encountered.
F6 13 00B1 335 BEQL   5$ ; Branch if no CRD error occurred.
1E 11 00B3 336 BRB    ERR_EXIT_8SS ; Else take error path.
00B5 337
00B5 338 .ALIGN LONG ; All handlers longword-aligned.
00B8 339
00B8 340 ;
00B8 341 ; Handler that gains control when a page has gross memory errors. Just
00B8 342 ; clear the error, recover the stack top, and advance to the next page.
00B8 343 ;
00B8 344 ;
00B8 345 PAGE_MCHECK_8SS: ; Handle machine check.
51    0100 C4  D0 00B8 346 MOVL   BIMEMSL_CSR1(R4),R1 ; Read memory CSR1.
0100 C4 51    D0 00BD 347 MOVL   R1,BIMEMSL_CSR1(R4) ; Re-write (write 1 to clear) CSR1.
51    0104 C4  D0 00C2 348 MOVL   BIMEMSL_CSR2(R4),R1 ; Read memory CSR2.
0104 C4 51    D0 00C7 349 MOVL   R1,BIMEMSL_CSR2(R4) ; Re-write (write 1 to clear) CSR2.
26    FFFFFFFF 8F DA 00CC 350 MTPR   #-1,#PR$_MCESR ; Clear error indicator.
FF2A' 31 00D3 351 ERR_EXIT_8SS:
00D6 352 BRW    BOO$PAGE_MCHECK ; Exit to common bad page code
00D6 353 .END
```

BTMEM8SS
Symbol table

- Configure and Test 11/8SS Memory M 11

15-SEP-1984 23:44:25 VAX/VMS Macro V04-00
4-SEP-1984 23:03:23 [BOOTS.SRC]BTMEM8SS.MAR;1Page 9
(3)

BIICSL_EAR	=	00000024		
BIICSL_SAR	=	00000020		
BIICSS_MEMNODE	=	00000007		
BIICSV_MEMNODE	=	00000008		
BIMEMSL_CSR1	=	00000100		
BIMEMSL_CSR2	=	00000104		
BIMEMSM_CRDLOGR	=	20000000		
BIMEMSV_BROKE	=	0000000C		
BOOSPAGE_MCHECK	*****		X	02
BOOSTEST_MEM	*****		X	02
CHECKMEM_8SS	00000000		RG	02
DMPSS_TR	=	00000008		
DMPSV_TR	=	00000018		
DO NEXT 8SS	0000006C		R	02
ERR_EXIT 8SS	000000D3		R	02
INIT_SEARCH	00000000		R	02
IO8SS\$AL_IOBASE	=	20000000		
IO8SS\$AL_NNEX	=	00000010		
IO8SS\$AL_PERNEX	=	00002000		
PAGE_MCHECK_8SS	000000B8		R	02
PR\$ MCSR	=	00000026		
PR8SS\$M_CADR_D	=	00000001		
PR8SS\$V_CADR_D	=	00000000		
PR8SS\$ CADR	=	00000025		
RPBSL_BOOTR5	=	00000030		
RPBSL_MEMDSC	=	000000BC		
RPBSM_FINDMEM	=	00004000		
RPBSM_MPM	=	00000800		
RPBSM_USEMPM	=	00001000		
RPBSV_CRDTEST	=	00000010		
TEST_QUAD_8SS	00000088		R	02
TRY_NEXUS_8SS	0000001D		R	02

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes															
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					
YBTMEM	000000D6 (214.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG					

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:01.81
Command processing	118	00:00:00.59	00:00:06.13
Pass 1	234	00:00:06.50	00:00:15.60
Symbol table sort	0	00:00:00.87	00:00:01.69
Pass 2	75	00:00:01.37	00:00:02.74
Symbol table output	5	00:00:00.05	00:00:00.05
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00

Assembler run totals 464 00:00:09.48 00:00:28.06

The working set limit was 1350 pages.
35196 bytes (69 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 631 non-local and 2 local symbols.
353 source lines were read in Pass 1, producing 13 object records in Pass 2.
18 pages of virtual memory were used to define 17 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	6
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	13

719 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:BTMEM8SS/OBJ=OBJ\$:BTMEM8SS MSRC\$:BTMEM8SS/UPDATE=(ENH\$:BTMEM8SS)+EXECML\$/LIB+LIB\$:BOOTS.MLB/LIB

0037 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

